# What I learned from LuaJIT

Excelsior JET

V8

Dart VM

LuaJIT

# torch

«A SCIENTIFIC COMPUTING FRAMEWORK FOR LUAJIT»

~~deep internal insight~~

overview of interesting things

```lua
local p = { x = 1, y = 1 }
for i = 1, 100 do
  p = { x = p.x + i,
        y = p.y - i }
end
```

## whirlwind introduction to Lua

```lua
-- dynamically typed
local v
v = 1
v = "string"
v = true
v = { } -- table
v = function () end
```

```lua
-- tables are key-value dictionaries
-- key is any type
local p = {
  x = 1,
  y = 1,
}
```

```lua
-- tables are key-value dictionaries
-- key is any type
local p = {
  ['x'] = 1,
  ['y'] = 1,
  [222] = 1,
  [{ }] = 1
}
```

```lua
-- single numeric type:
-- double precision floating point
type(1)   -- 'number'
type(1.0) -- 'number'
type(1.1) -- 'number'
```

```lua
-- metatables alter behavior of tables
local tbl = {}
setmetatable(tbl, {
  __index = function (self, key)
    print('index', key)
    return 0
  end,
  __newindex = function (self, key, val)
    print('newindex', key, val)
  end
})
```

```
-- metatables alter behavior of tables
print(tbl['somekey'])
-- index somekey
-- 0
tbl[42] = 'somevalue';
-- newindex 42 somevalue
```

```lua
local tbl = {}
setmetatable(tbl, {
  __index = { x = 42 }
})
print(tbl.x) -- 42
```

```
-- metatables alter behavior of tables
setmetatable(tbl, {
  -- will be called when evaluating
  -- + expression with tbl
  __add = function ()
    ...
  end
})
```

```
local p = { x = 1, y = 1 }
for i = 1, 100 do
  p = { x = p.x + i,
        y = p.y - i }
end
```

```
->LOOP:
  xorps xmm5, xmm5
  cvtsi2sd xmm5, ebp
  addsd xmm6, xmm5
  subsd xmm7, xmm5
  add ebp, +0x01
  cmp ebp, +0x64
  jle ->LOOP
  jmp ->4
```

# « how does it do it? »

learning by reading sources

```lua
local p = { x = 1, y = 1, [1] = 1 }
for i = 1, 100 do
  p = { x = p.x + i,
        y = p.y - i,
        [1] = p[1] }
end
```

```
->LOOP:
  movsd [rsp+0x28], xmm6
  movsd [rsp+0x30], xmm7
  mov [rsp+0x24], eax
  mov edi, [0x000423d8]
  cmp edi, [0x000423dc]
  jb skip
  mov esi, 0x1
  mov edi, 0x000423b8
  call ->lj_gc_step_jit
  test eax, eax
  jnz ->4
skip:
  mov edi, [0x000424b0]
  mov esi, 0x00052948
  call ->lj_tab_dup
  mov esi, eax
  mov [rsp+0x20], esi
  mov edi, [0x000424b0]
  mov eax, [rsp+0x24]
  movsd xmm7, [rsp+0x30]
  movsd xmm5, [rsp+0x28]
  cmp dword [rax+0x1c], +0x01
  jnz ->4
  mov r15d, [rax+0x14]
  mov rbx, 0xfffffffb00053e50
  cmp rbx, [r15+0x20]
  jnz ->4
```

```
  xorps xmm6, xmm6
  cvtsi2sd xmm6, ebp
  addsd xmm5, xmm6
  movsd [rsp+0x10], xmm5
  mov ebx, [rsi+0x14]
  movsd [rbx+0x18], xmm5
  mov rdx, 0xfffffffb0004a188
  cmp rdx, [r15+0x8]
  jnz ->5
  subsd xmm7, xmm6
  movsd [rsp+0x18], xmm7
  movsd [rbx], xmm7
  cmp dword [rax+0x18], +0x01
  jbe ->6
  mov ebx, [rax+0x8]
  cmp dword [rbx+0xc], 0xfffeffff
  jnb ->6
  movsd xmm5, [rbx+0x8]
  movsd [rsp+0x8], xmm5
  mov edx, 0x000535d8
  call ->lj_tab_newkey
  mov ebx, eax
  mov eax, [rsp+0x20]
  movsd xmm7, [rsp+0x18]
  movsd xmm6, [rsp+0x10]
  movsd xmm5, [rsp+0x8]
  movsd [rbx], xmm5
  add ebp, +0x01
  cmp ebp, +0x64
  jle ->LOOP
  jmp ->7
```

## « why does it **not** do it? »

learning by fixing bugs

# 1GB memory limit

(pre v2.1)

## Lua is dynamically typed

## NaN-tagging

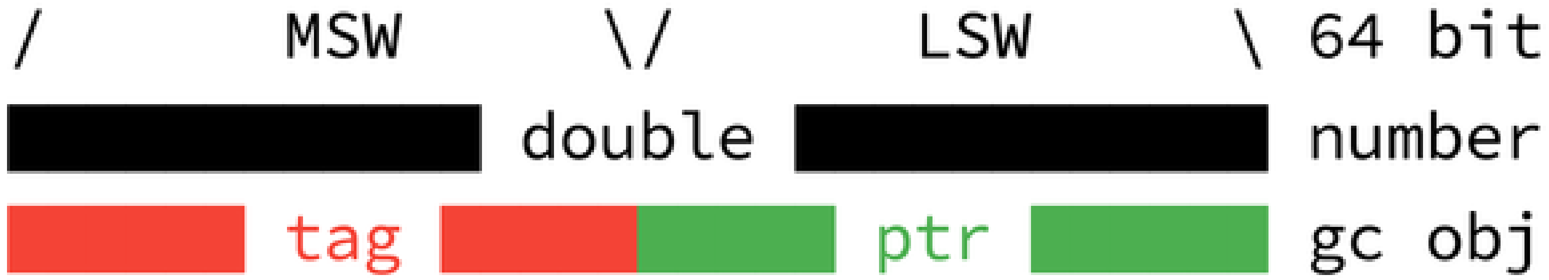
```
sign      mantissa (52 bit)
v     /------------------------\
████████████████████████████████
 \---/
  exponent (11 bit)
```

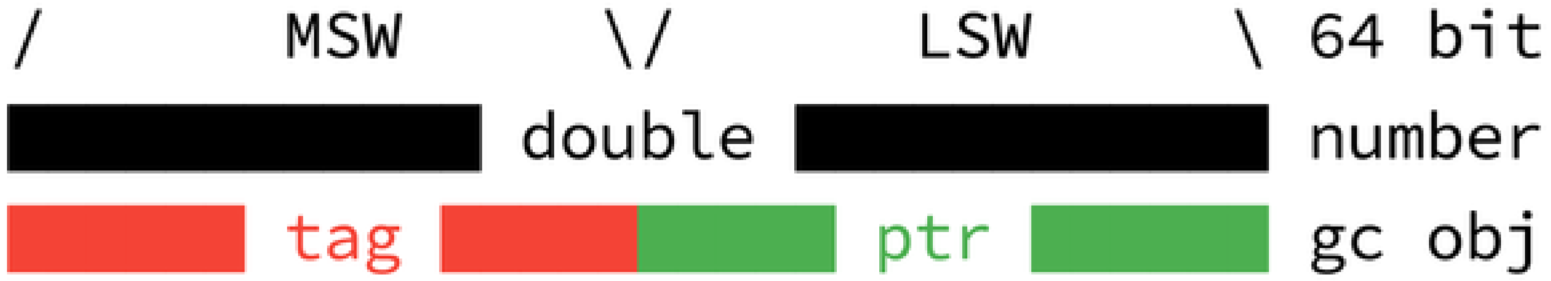
```
sign      mantissa (52 bit)
v     /------------------------\
████████████████████████████████
 \---/
  exponent (11 bit)

NaN: E = 7ff & M ≠ 0
```

```
sign        mantissa (52 bit)
v      /------------------------\
████████████████████████████████ (64 bits)
 \---/
  exponent (11 bit)

NaN: E = 7ff & M ≠ 0 (whole family of NaNs)
```

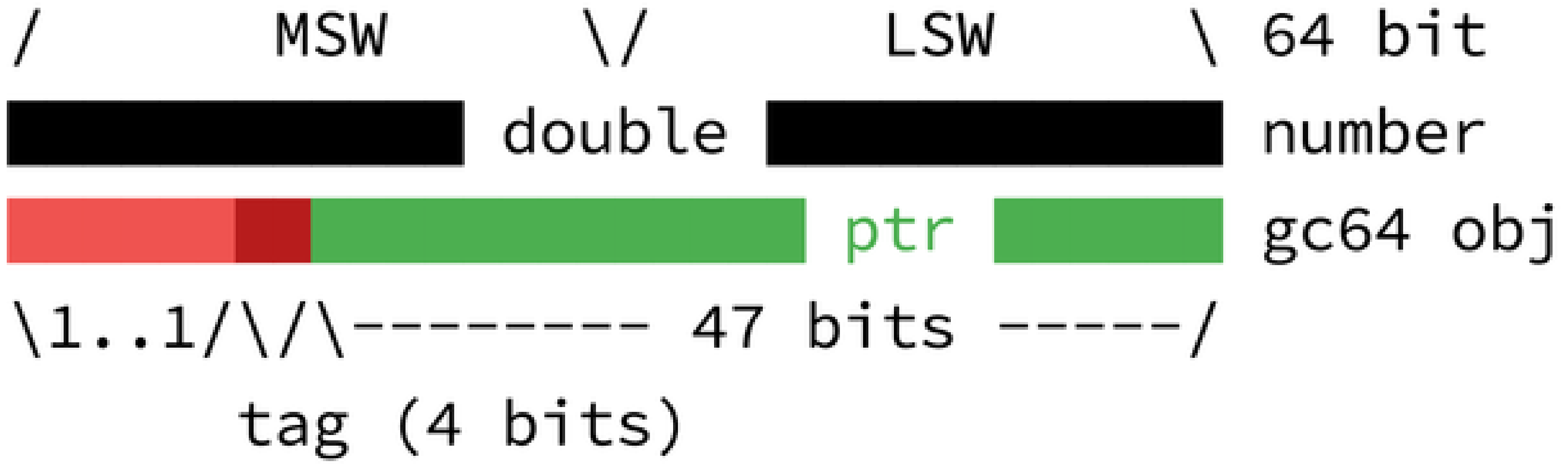
## `TValue`

dynamically typed slot


/ MSW \/ LSW \ 64 bit
double
number
tag
ptr
gc obj



```
number tag < ffff0000
table  tag = fffffff4 = ~11u
```


/ MSW \/ LSW \ 64 bit
double
number
ptr
gc64 obj
\1..1/\/\--------- 47 bits -----/
tag (4 bits)

## kinda works

AArch64: 52-bit VA

changing tagging
tough exercise

```
...
| 
| // Macros to test operand types.
| .macro checktp, reg, tp
|   cmp dword [BASE+reg*8+4], tp
| .endmacro
| .macro checktab, reg, target
|   checktp reg, LJ_TTAB
|   jne target
| .endmacro
| 
...
  case BC_TGETB:
    |   ins_ABC  // RA = dst, RB = table, RC = byte literal
    |   checktab RB, ->vmeta_tgetb
    |   mov TAB:RB, [BASE+RB*8]
...
```

## DynASM

generates code that
generates code

```
case BC_TGETB:
    //|  ins_ABC  // RA = dst, RB = table, RC = byte literal
    //|  checktab RB, ->vmeta_tgetb
    //|  mov TAB:RB, [BASE+RB*8]
...
    dasm_put(Dst, 10994, LJ_TTAB, Dt6(->asize), Dt6(->array), LJ_TNIL,
// Type definitions. Some of these are only used for documentation.
.type L,     lua_State
.type GL,    global_State
...
  |  mov GL:RB, L:RB->glref
  |  mov dword GL:RB->vmstate, ~LJ_VMST_C
```

```
|// Type definitions. Some of these are only used for documentation.
|.type L,    lua_State
|.type GL,   global_State
...
  |   mov GL:RB, [RB, #offsetof(lua_State, glref)]
  |   mov dword GL:RB->vmstate, ~LJ_VMST_C
```

no actual understanding of types

```
|  cmp dword L:RB->openupval, 0
|   cmp dword L:RB->openupval, 0
              ^^^^^^^^^^^^^^^^ pointer
```

```
|   cmp aword L:RB->openupval, 0
```

## what is interpreter interpreting?

```
/----- 32 bits -----\
+----+----+----+----+
| OP |    |    |    | Format ABC
+----+----+----+----+
| OP |    |         | Format AD
+---------+----+----+
0...................32
```

```
     BASE
     ↓
~----+-----+-----+-----+-----+---~
     | R0  | R1  | R2  | R3  |
~----+-----+-----+-----+-----+---~
      ↑↑↑↑↑
      TValue (64bit)
```

```
CALL A, ResN, ArgN
                          F <- R(A);
R(A), ..., R(A+ResN-2) <- F(R(A+1), ..., R(A+ArgN-1)), if ResN != 0
R(A), ...              <- F(R(A+1), ..., R(A+ArgN-1)), if ResN == 0
```

```
     BASE
     ↓
~----+-~ ~-+-------+-------+-------+---~
     |     | Func  | Arg0  | Arg1  |
~----+-~ ~-+-------+-------+-------+---~
           ↑
           R(A)
     BASE         BASE
     ↓            ↓
~----+-~ ~-+------+------+------+---~
     |     | Func | Arg0 | Arg1 |
~----+-~ ~-+------+------+------+---~
                  ↑
                  R(0)
```

## frame linking

```
     BASE           BASE

      ↓              ↓
~-----+-~  ~-+-------+-------+-------+---~
      |      | Func  | Arg0  | Arg1  |
~-----+-~  ~-+-------+-------+-------+---~
            /         \
           /            \
        [   tag | ptr   ]
      BASE           BASE
      ↓              ↓
~-----+-~  ~-+-------+-------+-------+---~
      |      | Func  | Arg0  | Arg1  |
~-----+-~  ~-+-------+-------+-------+---~
            /         \
           /           \
        [ link  | ptr   ]
```

```
         link |
--------------+-------------------
      PC   00 | Lua frame
   delta 001 | C frame
   delta 010 | Continuation frame
   delta 011 | Lua vararg frame
   delta 101 | cpcall() frame
        .... etc ...

PC is 4 byte aligned
delta is 8 byte aligned
```

```
         link |
--------------+---------------------
       PC  00 | Lua frame
   delta 001 | C frame
   delta 010 | Continuation frame
   delta 011 | Lua vararg frame
   delta 101 | cpcall() frame
        .... etc ...

PC is 4 byte aligned
delta is 8 byte aligned
```

## when unwinding look at PC-1 to determine caller's BASE

```
CALL A, ... => CallerBASE = BASE - A
```

```
         link |
--------------+--------------------
      PC   00 | Lua frame
   delta  001 | C frame
   delta  010 | Continuation frame
   delta  011 | Lua vararg frame
   delta  101 | cpcall() frame
        .... etc ...

PC is 4 byte aligned
delta is 8 byte aligned
```

*continuations* allow to specify action to perform when callee returns

```
;; jump to target if R(A) == R(D)
ISEQV A, D
JUMP target
```

```
;; jump to target if R(A) == R(D)
ISEQV A, D
JUMP target
;; what if R(A) has __eq metamethod?
;; jump to target if R(A) == R(D)
ISEQV A, D
JUMP target
;; what if R(A) has __eq metamethod?
;; need to call metamethod
;; ... then branch on return
;; jump to target if R(A) == R(D)
ISEQV A, D
JUMP target
;; what if R(A) has __eq metamethod?
;; need to call metamethod
;; ... then branch on return
```

```
interpreter
+--------------------+
|      ...           |
| PC → ISEQV A, D    |
|      JUMP target   |
|      ...           |
+--------------------+
interpreter
+--------------------+
| +--------------------+
| | nested interpreter |
| | for the metamethod |
| |                    |
+-|                    |
  +--------------------+
interpreter
+---------------------+
|       ...           | branch on the result from
| PC → ISEQV A, D     | the nested interpreter
|       JUMP target   |
|       ...           |
+---------------------+
```

continuations make it simpler

```
     BASE                        metamethod
     ↓                          /-- frame -->
~----+-~ ~-+------+------+------+------+---~
     |     |      |___   |      |      |
~----+-~ ~-+------+-↑----+------+------+---~
     \------------/ continuation callback
      current frame   (e.g. cont_condt)
```

## let's talk about DISPATCH

```
|  jmp aword [DISPATCH+OP*4]
|  jmp aword [DISPATCH+OP*4]
              ↑
              can replace handlers
```

- hooks (~ debugging)
- profiling
- recording

```
;; hotcounting
;; loop bytecodes
FORL
ITERL
LOOP

;; function entries
FUNCF
```

```
|.macro hotloop, reg
|  mov reg, PC
|  shr reg, 1
|  and reg, HOTCOUNT_PCMASK
|  sub word [DISPATCH+reg+GG_DISP2HOT],
|            HOTCOUNT_LOOP
|  jb ->vm_hotloop
|.endmacro
```

```
hotcount[(PC>>2) & (HOTCOUNT_SIZE-1)]
```

```
#define HOTCOUNT_SIZE   64

hotcount[(PC>>2) & (HOTCOUNT_SIZE-1)]
```

```
#define HOTCOUNT_SIZE   64

hotcount[(PC>>2) & (HOTCOUNT_SIZE-1)]

/* can cause non-determenism */
```

## recording pipeline

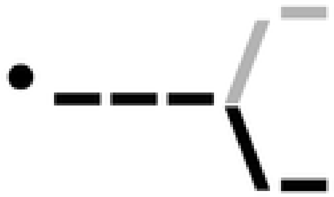
## tracing 101

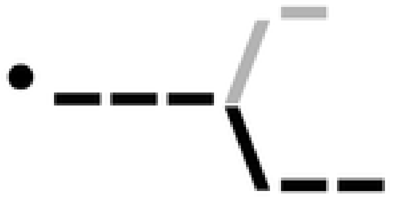
-

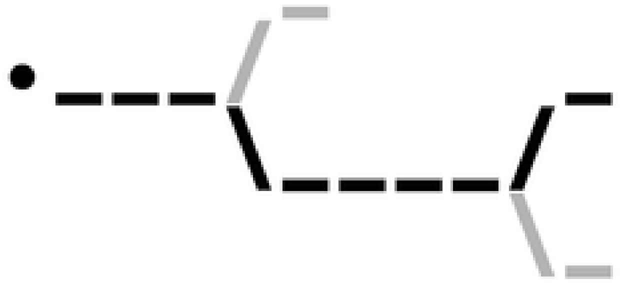
- _

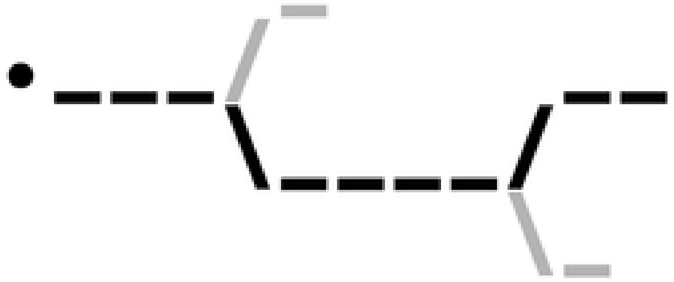
- __

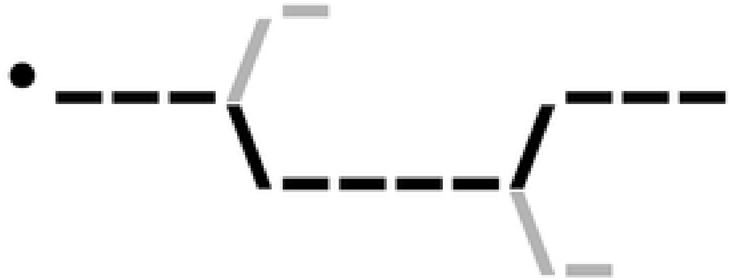
- ___

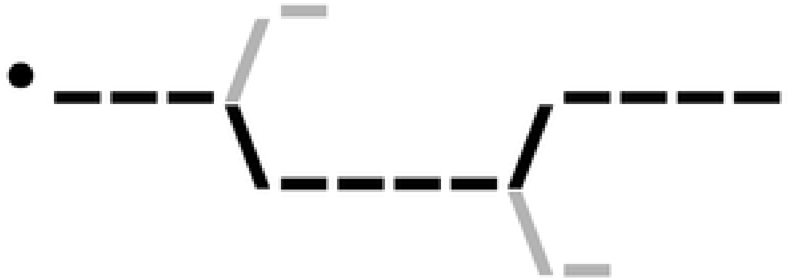


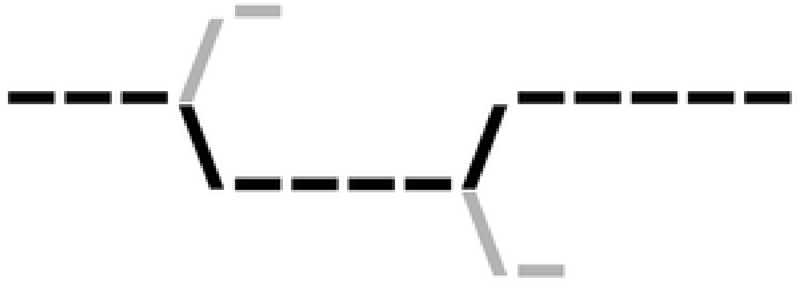


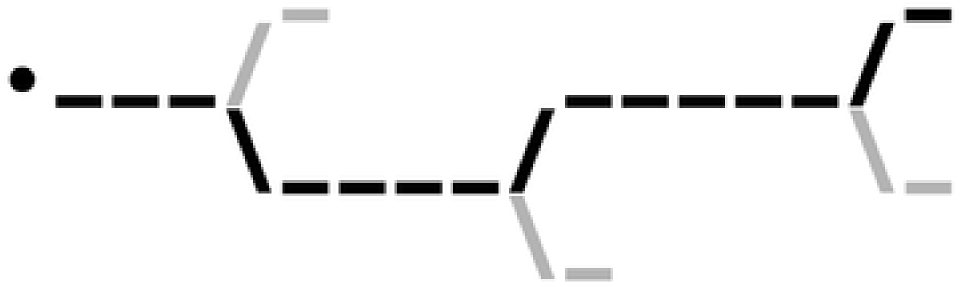


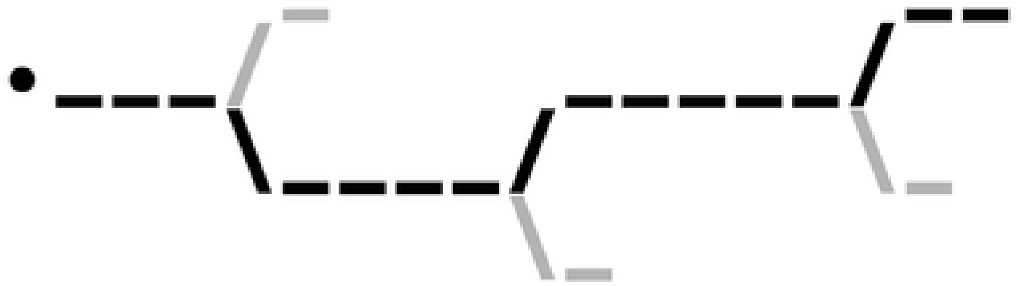


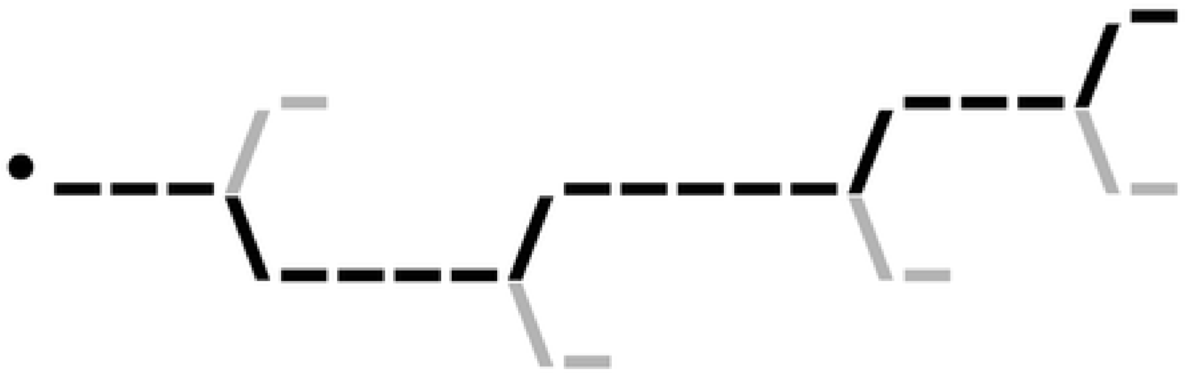


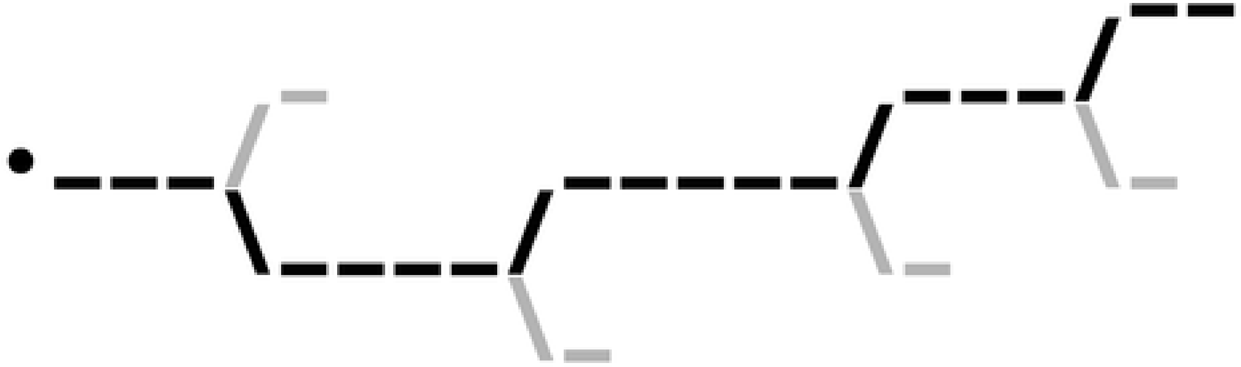


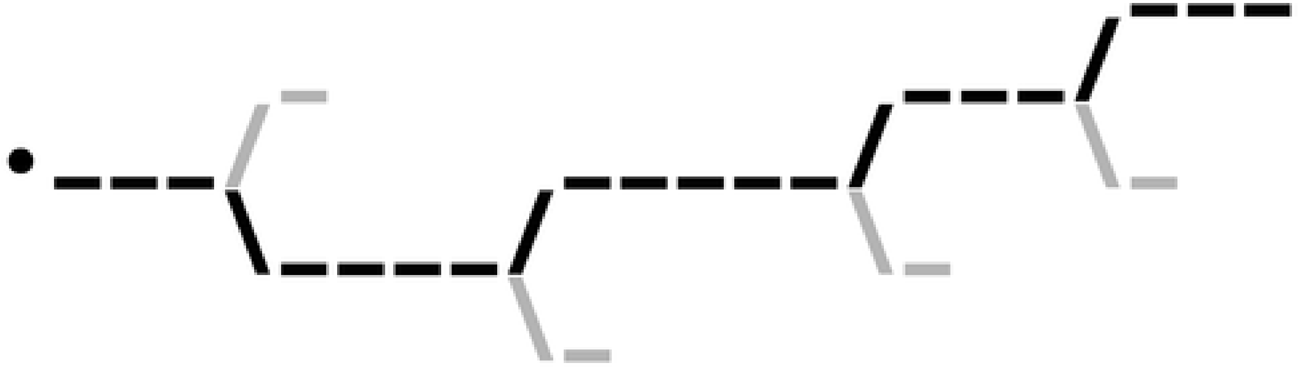
- 

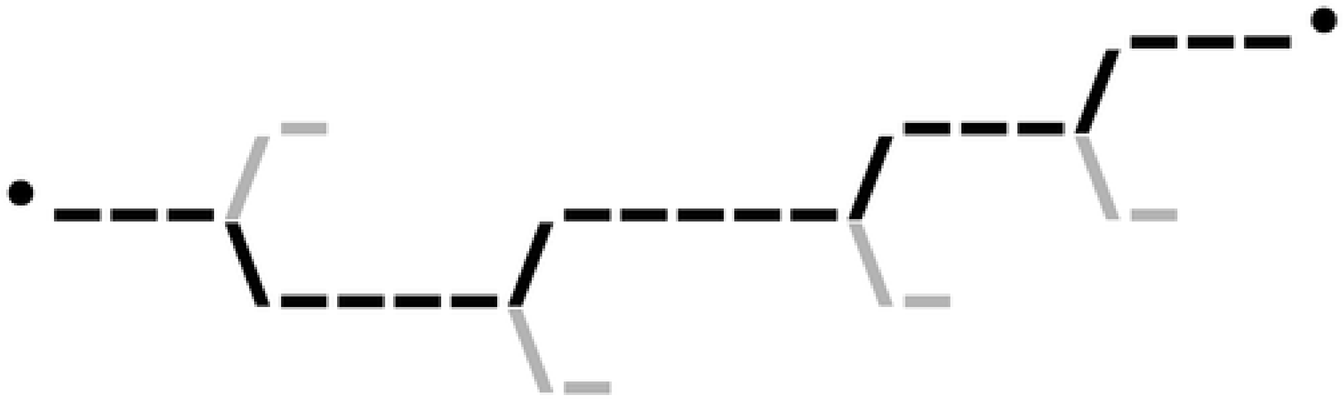


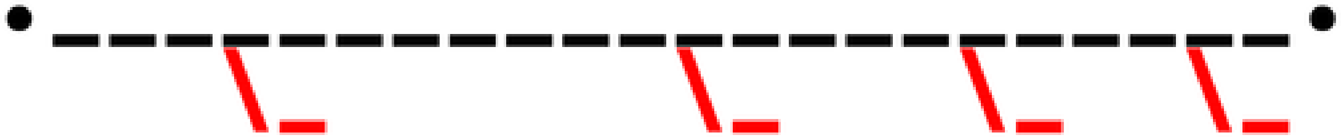


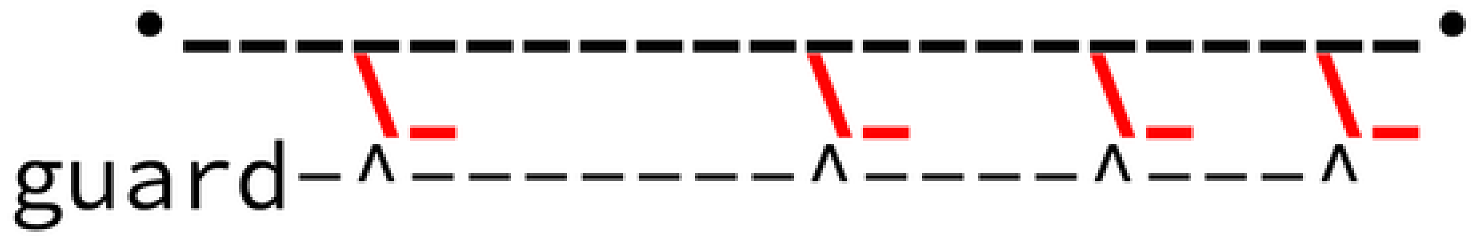




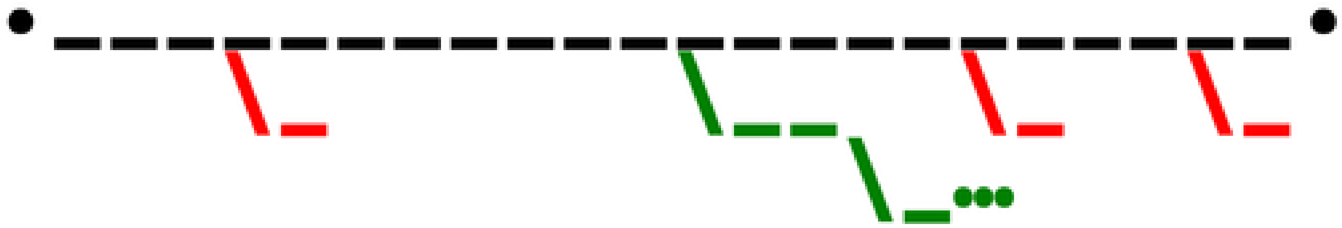







guard-^--------^-----^----^

## hot side exits spawn side traces



back to recording

```
            concrete                  SSA
             values                   refs
INTERPRETER    |           RECORDER    |
+--------------v------+    +-----------v-------+
|  ~-+---+---+---+-~  |    | ~-+---+---+---+-~ |
|    |   |   |   |    |    |   |   |   |   |   |
|  ~-+---+---+---+-~  |    | ~-+---+---+---+-~ |
|  ...                |    |               ... |
|> ADDVV r0, r0, r1 ==================> SSA IR |
|  ...                |    |               ... |
+---------------------+    +-------------------+
            concrete                  SSA
             values                   refs
INTERPRETER    |           RECORDER    |
+--------------v------+    +-----------v-------+
|  ~-+---+---+---+-~  |    | ~-+---+---+---+-~ |
|    |num|num|   |    |    |   |001|002|   |   |
|  ~-+---+---+---+-~  |    | ~-+---+---+---+-~ |
|  ...                |    |               ... |
|> ADDVV r0, r0, r1 ==================> SSA IR |
|  ...                |    |               ... |
+---------------------+    +-------------------+
                 concrete                                SSA
                  values                                 refs
INTERPRETER         |               RECORDER              |
+-------------------v-------+      +-----------------v--------+
|   ~-+---+---+---+-~       |      | ~-+---+---+---+-~        |
|     |num|num|   |         |      |   |003|002|   |          |
|   ~-+---+---+---+-~       |      | ~-+---+---+---+-~        |
|   ADDVV r0, r0, r1        |      | 003: ADD 001, 002        |
|>  SUBVN r1, r1, +1 ==================> SSA IR               |
|   ...                     |      |                    ...   |
+---------------------------+      +--------------------------+
                concrete                             SSA
                 values                              refs
INTERPRETER        |              RECORDER            |
+------------------v-------+      +--------------v--------+
|   ~-+---+---+---+-~      |      | ~-+---+---+---+-~     |
|     |num|num|   |        |      |   |003|004|   |       |
|   ~-+---+---+---+-~      |      | ~-+---+---+---+-~     |
|   SUBVN r1, r1, +1       |      | 004: SUB 002,  +1     |
|>  ...                 ======================> SSA IR    |
|   ...                    |      |                 ...   |
+--------------------------+      +-----------------------+
```

# IR

```c
/* Trace object. */
typedef struct GCtrace {
  /* IR instructions/constants.
  ** Biased with REF_BIAS.
  */
  IRIns *ir;

} GCtrace;
```

```
/* Trace object. */
typedef struct GCtrace {
  /* IR instructions/constants.
  ** Biased with REF_BIAS.
  */
  IRIns *ir;

} GCtrace;
```

```c
typedef uint16_t IRRef1;

/* Fixed references. */
enum {
  REF_TRUE =  REF_BIAS-3,
  REF_FALSE = REF_BIAS-2,
  REF_NIL = REF_BIAS-1,
  /* \--- Constants grow downwards. */
  REF_BIAS =  0x8000,
  /* /--- IR grows upwards. */
  REF_FIRST = REF_BIAS+1,
  REF_DROP =  0xffff
};
```

```
    <-- constants --\ /-- non-constants -->
~--+-----+-----+-----+-----+-----+-----+--~
   |false|true |nil  |     |     |     |
~--+-----+-----+-----+-----+-----+-----+--~
                     ^ &ir[REF_BIAS]

ir := irbuf + nconsts - REF_BIAS
```

## IRIns

```
   16      16      8   8   8   8
+-------+-------+---+---+---+---+
|  op1  |  op2  | t | o | r | s |
+-------+-------+---+---+---+---+
|  op12/i/gco   |   ot  | prev  |
+---------------+-------+-------+
       32           16      16
```

```
+-------+-------+---+---+---+---+
|  op1  |  op2  | t | o | r | s |
+-------+-------+---+---+---+---+
|  op12/i/gco   |   ot  | prev  |
+---------------+-------+-------+
```

```
prev is the reference to the previous
     instruction with the same opcode
```

```
+--------+--------+---+---+---+---+
|  op1   |  op2   | t | o | r | s |
+--------+--------+---+---+---+---+
|  op12/i/gco     |   ot  | prev  |
+-----------------+-------+-------+
```

**r/s** register allocation state

```
+--------+--------+---+---+---+---+
|  op1   |  op2   | t | o | r | s |
+--------+--------+---+---+---+---+
|  op12/i/gco     |   ot  | prev  |
+-----------------+-------+-------+
```

**o** opcode
**t** type

```
+--------+--------+---+---+---+---+
|  op1   |  op2   | t | o | r | s |
+--------+--------+---+---+---+---+
|  op12/i/gco     |   ot  | prev  |
+-----------------+-------+-------+
```

**op1/op2** IR references

```
+--------+--------+---+---+---+---+
|  op1   |  op2   | t | o | r | s |
+--------+--------+---+---+---+---+
|  op12/i/gco     |   ot  | prev  |
+-----------------+-------+-------+
```

**i/gco** constants (32 bit)

```
/* Tagged IR references (32 bit).
**
** +-------+-------+---------------+
** |  irt  | flags |      ref      |
** +-------+-------+---------------+
**
** The tag holds a copy of the IRType
** and speeds up IR type checks.
*/

typedef uint32_t TRef;
```

```
BYTECODE ======================> SSA IR
BYTECODE ===================> SSA IR
|                                  ^
v                                  |
interpret -> specialize -> fold&emit
```

```c
case BC_LEN:
  if (tref_isstr(rc))
    rc = emitir(IRTI(IR_FLOAD), rc, IRFL_STR_LEN);
  else if (!LJ_52 && tref_istab(rc))
    rc = lj_ir_call(J, IRCALL_lj_tab_len, rc);
  else
    rc = rec_mm_len(J, rc, rcv);
  break;
```

```
case BC_LEN:
  if (tref_isstr(rc))
    rc = emitir(IRTI(IR_FLOAD), rc, IRFL_STR_LEN);
  else if (!LJ_52 && tref_istab(rc))
    rc = lj_ir_call(J, IRCALL_lj_tab_len, rc);
  else
    rc = rec_mm_len(J, rc, rcv);
  break;
```

```
case BC_LEN:
  if (tref_isstr(rc))
    rc = emitir(IRTI(IR_FLOAD), rc, IRFL_STR_LEN);
  else if (!LJ_52 && tref_istab(rc))
    rc = lj_ir_call(J, IRCALL_lj_tab_len, rc);
  else
    rc = rec_mm_len(J, rc, rcv);
  break;
```

`emitir` passes instruction to FOLD engine

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */

  return fleft->op2;
}
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */

  return fleft->op2;
}
// Rules hashtable generated by build
// Rules applied until fixpoint
```

```
FWD
DSE
NARROW
ABCelim
CSE
```

```
DCE
LOOP
SPLIT
SINK
```

DCE
LOOP
SPLIT
SINK

```lua
local sum = 0
for i = 1, n do
  sum = sum + arr[i]
end
```

```
0006 TGETV  r8, r1, r7
0007 ADDVV  r3, r3, r8
0008 FORL   r4 => 0006
```

```
0006 TGETV  r8, r1, r7 ; r8 = r1[r7]
0007 ADDVV  r3, r3, r8 ; r3 = r3 + r8
0008 FORL   r4 => 0006 ; r4 = r4 + r6
                       ; if r4 <= r5 then
                       ;   r7 = r4
                       ;   jump 0006
                       ; end
```

```
                   arr       sum  (i)  lim  step i
              R0   R1   R2   R3   R4   R5   R6   R7   R8
       [ ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ]
                       ‖
0006 TGETV  r8, r1, r7 ‖
0007 ADDVV  r3, r3, r8 ‖
0008 FORL   r4 => 0006 ‖
                       ‖
                       ‖
                       ‖
                       ‖
                       ‖
                       ‖
                       ‖
                       ‖
                       ‖
```

```
                    arr            sum  (i)  lim  step i
                R0  R1   R2   R3   R4   R5   R6   R7   R8
          [ ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ]
⇒ 0005 FORI   r4 => 0009 ‖
  0006 TGETV  r8, r1, r7 ‖
  0007 ADDVV  r3, r3, r8 ‖
  0008 FORL   r4 => 0006 ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                   arr          sum   (i)   lim   step i
              R0    R1    R2    R3    R4    R5    R6    R7    R8
          [ ---- ---- ---- ---- ---- 0003 0001 ---- 0003 ---- ]
⇒ 0005 FORI    r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV   r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV   r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL    r4 => 0006 ‖
                          ‖
```

```
                     arr          sum  (i)  lim  step i
                R0   R1   R2   R3   R4   R5   R6   R7   R8
        [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 ---- ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
⇒ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                    arr         sum  (i)  lim  step i
               R0   R1   R2   R3   R4   R5   R6   R7   R8
         [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 ---- ]
  0005 FORI   r4 => 0009 ║ 0001 SLOAD  R5
⇒ 0006 TGETV  r8, r1, r7 ║ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ║ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ║ 0004 SLOAD  R1
                         ║ 0005 FLOAD  0004  tab.asize
                         ║
                         ║
                         ║
                         ║
                         ║
                         ║
                         ║
                         ║
```

```
                  arr       sum  (i)  lim  step i
            R0   R1   R2   R3   R4   R5   R6   R7   R8
        [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 ---- ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
⇒ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                    arr         sum  (i)  lim  step i
               R0   R1   R2   R3   R4   R5   R6   R7   R8
          [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 ---- ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
⇒ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                     arr         sum  (i)  lim  step i
                 R0   R1   R2   R3   R4   R5   R6   R7   R8
          [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 ---- ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
⇒ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                    arr         sum  (i)  lim  step i
             R0    R1    R2    R3    R4    R5    R6    R7    R8
        [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 0009 ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
⇒ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                   arr         sum  (i)  lim  step i
              R0   R1   R2   R3   R4   R5   R6   R7   R8
         [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 0009 ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
⇒ 0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                     arr         sum  (i)  lim  step i
                R0   R1    R2    R3   R4   R5   R6   R7   R8
          [ ---- ---- 0004 ---- 0010 0003 0001 ---- 0003 0009 ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
⇒ 0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖ 0010 SLOAD  R3
                         ‖
                         ‖
                         ‖
```

```
                    arr         sum   (i)   lim   step i
              R0    R1    R2    R3    R4    R5    R6    R7    R8
      [ ---- ---- 0004 ---- 0011 0003 0001 ---- 0003 0009 ]
  0005 FORI    r4 => 0009 ║ 0001 SLOAD  R5
  0006 TGETV   r8, r1, r7 ║ 0002 LE     0001  +2147483646
⇒ 0007 ADDVV   r3, r3, r8 ║ 0003 SLOAD  R4
  0008 FORL    r4 => 0006 ║ 0004 SLOAD  R1
                          ║ 0005 FLOAD  0004  tab.asize
                          ║ 0006 ABC    0005  0001
                          ║ 0007 FLOAD  0004  tab.array
                          ║ 0008 AREF   0007  0003
                          ║ 0009 ALOAD  0008
                          ║ 0010 SLOAD  R3    T
                          ║ 0011 ADD    0010  0009
                          ║
                          ║
                    arr          sum   (i)   lim   step i
             R0     R1     R2     R3     R4     R5     R6     R7     R8
      [ ---- ---- 0004 ---- 0011 0003 0001 ---- 0003 0009 ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
⇒ 0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖ 0010 SLOAD  R3
                         ‖ 0011 ADD    0010  0009
                         ‖
                         ‖
                    arr         sum  (i)  lim  step i
              R0    R1    R2    R3    R4    R5    R6    R7    R8
        [ ---- ---- 0004 ---- 0011 0012 0001 ---- 0012 0009 ]
  0005 FORI   r4 => 0009 || 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 || 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 || 0003 SLOAD  R4
⇒ 0008 FORL   r4 => 0006 || 0004 SLOAD  R1
                         || 0005 FLOAD  0004  tab.asize
                         || 0006 ABC    0005  0001
                         || 0007 FLOAD  0004  tab.array
                         || 0008 AREF   0007  0003
                         || 0009 ALOAD  0008
                         || 0010 SLOAD  R3
                         || 0011 ADD    0010  0009
                         || 0012 ADD    0003  +1
                         ||
                     arr        sum  (i)  lim  step i
                R0   R1   R2    R3   R4   R5   R6   R7   R8
        [ ---- ---- 0004 ---- 0011 0012 0001 ---- 0012 0009 ]
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
⇒ 0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖ 0010 SLOAD  R3
                         ‖ 0011 ADD    0010  0009
                         ‖ 0012 ADD    0003  +1
                         ‖ 0013 LE     0012  0001
```

```
0001 >  int SLOAD  #6    CRI
0002 >  int LE     0001  +2147483646
0003    int SLOAD  #5    CI
0004 >  tab SLOAD  #2    T
0005    int FLOAD  0004  tab.asize
0006 >  p32 ABC    0005  0001
0007    p32 FLOAD  0004  tab.array
0008    p32 AREF   0007  0003
0009 >  num ALOAD  0008
0010 >  num SLOAD  #4    T
0011  + num ADD    0010  0009
0012  + int ADD    0003  +1
0013 >  int LE     0012  0001
....        SNAP   #2   [ - - - - 0011 0012 0001 - 0012 ]
```

```
0001 >   int SLOAD  #6    CRI
0002 >   int LE     0001  +2147483646
0003     int SLOAD  #5    CI
0004 >   tab SLOAD  #2    T
0005     int FLOAD  0004  tab.asize
0006 >   p32 ABC    0005  0001
0007     p32 FLOAD  0004  tab.array
0008     p32 AREF   0007  0003
0009 >   num ALOAD  0008
0010 >   num SLOAD  #4    T
0011   + num ADD    0010  0009
0012   + int ADD    0003  +1
0013 >   int LE     0012  0001
....         SNAP   #2   [ - - - - 0011 0012 0001 - 0012 ]
```

```
0001 SLOAD  #6    CRI          |
0002 LE     0001  +2147483646  |
0003 SLOAD  #5    CI           |
0004 SLOAD  #2    T            |
0005 FLOAD  0004  tab.asize    |
0006 ABC    0005  0001         |
0007 FLOAD  0004  tab.array    |
0008 AREF   0007  0003         |
0009 ALOAD  0008               |
0010 SLOAD  #4    T            |
0011 ADD    0010  0009         |
0012 ADD    0003  +1           |
0013 LE     0012  0001         |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6   CRI         |
0002 LE     0001 +2147483646 |
0003 SLOAD  #5   CI          |
0004 SLOAD  #2   T           |
0005 FLOAD  0004 tab.asize   |
0006 ABC    0005 0001        |
0007 FLOAD  0004 tab.array   |
0008 AREF   0007 0003        |
0009 ALOAD  0008             |
0010 SLOAD  #4   T           |
0011 ADD    0010 0009        |
0012 ADD    0003 +1          |
0013 LE     0012 0001        |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6   CRI           | ===> 0001
0002 LE     0001 +2147483646   |
0003 SLOAD  #5   CI            |
0004 SLOAD  #2   T             |
0005 FLOAD  0004 tab.asize     |
0006 ABC    0005 0001          |
0007 FLOAD  0004 tab.array     |
0008 AREF   0007 0003          |
0009 ALOAD  0008               |
0010 SLOAD  #4   T             |
0011 ADD    0010 0009          |
0012 ADD    0003 +1            |
0013 LE     0012 0001          |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI          | 0001
0002 LE     0001  +2147483646  |
0003 SLOAD  #5    CI           |
0004 SLOAD  #2    T            |
0005 FLOAD  0004  tab.asize    |
0006 ABC    0005  0001         |
0007 FLOAD  0004  tab.array    |
0008 AREF   0007  0003         |
0009 ALOAD  0008               |
0010 SLOAD  #4    T            |
0011 ADD    0010  0009         |
0012 ADD    0003  +1           |
0013 LE     0012  0001         |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI               | 0001
0002 LE     0001  +2147483646       | ===> LE    [0001] +2147483646
0003 SLOAD  #5    CI                |
0004 SLOAD  #2    T                 |
0005 FLOAD  0004  tab.asize         |
0006 ABC    0005  0001              |
0007 FLOAD  0004  tab.array         |
0008 AREF   0007  0003              |
0009 ALOAD  0008                    |
0010 SLOAD  #4    T                 |
0011 ADD    0010  0009              |
0012 ADD    0003  +1                |
0013 LE     0012  0001              |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI               | 0001
0002 LE     0001  +2147483646       | ===> LE     0001  +2147483646
0003 SLOAD  #5    CI                |
0004 SLOAD  #2    T                 |
0005 FLOAD  0004  tab.asize         |
0006 ABC    0005  0001              |
0007 FLOAD  0004  tab.array         |
0008 AREF   0007  0003              |
0009 ALOAD  0008                    |
0010 SLOAD  #4    T                 |
0011 ADD    0010  0009              |
0012 ADD    0003  +1                |
0013 LE     0012  0001              |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             |
0004 SLOAD  #2    T              |
0005 FLOAD  0004  tab.asize      |
0006 ABC    0005  0001           |
0007 FLOAD  0004  tab.array      |
0008 AREF   0007  0003           |
0009 ALOAD  0008                 |
0010 SLOAD  #4    T              |
0011 ADD    0010  0009           |
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI           | 0001
0002 LE     0001  +2147483646   | 0002
0003 SLOAD  #5    CI            | 0012
0004 SLOAD  #2    T             |
0005 FLOAD  0004  tab.asize     |
0006 ABC    0005  0001          |
0007 FLOAD  0004  tab.array     |
0008 AREF   0007  0003          |
0009 ALOAD  0008                |
0010 SLOAD  #4    T             |
0011 ADD    0010  0009          |
0012 ADD    0003  +1            |
0013 LE     0012  0001          |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      |
0006 ABC    0005  0001           |
0007 FLOAD  0004  tab.array      |
0008 AREF   0007  0003           |
0009 ALOAD  0008                 |
0010 SLOAD  #4    T              |
0011 ADD    0010  0009           |
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | ===> FLOAD  0004  tab.asize
0006 ABC    0005  0001           |
0007 FLOAD  0004  tab.array      |
0008 AREF   0007  0003           |
0009 ALOAD  0008                 |
0010 SLOAD  #4    T              |
0011 ADD    0010  0009           |
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI          | 0001
0002 LE     0001  +2147483646  | 0002
0003 SLOAD  #5    CI           | 0012
0004 SLOAD  #2    T            | 0004
0005 FLOAD  0004  tab.asize    | 0005
0006 ABC    0005  0001         |
0007 FLOAD  0004  tab.array    |
0008 AREF   0007  0003         |
0009 ALOAD  0008               |
0010 SLOAD  #4    T            |
0011 ADD    0010  0009         |
0012 ADD    0003  +1           |
0013 LE     0012  0001         |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI          | 0001
0002 LE     0001  +2147483646  | 0002
0003 SLOAD  #5    CI           | 0012
0004 SLOAD  #2    T            | 0004
0005 FLOAD  0004  tab.asize    | 0005
0006 ABC    0005  0001         | ===> ABC    0005  0001
0007 FLOAD  0004  tab.array    |
0008 AREF   0007  0003         |
0009 ALOAD  0008               |
0010 SLOAD  #4    T            |
0011 ADD    0010  0009         |
0012 ADD    0003  +1           |
0013 LE     0012  0001         |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI           | 0001
0002 LE     0001  +2147483646   | 0002
0003 SLOAD  #5    CI            | 0012
0004 SLOAD  #2    T             | 0004
0005 FLOAD  0004  tab.asize     | 0005
0006 ABC    0005  0001          | 0006
0007 FLOAD  0004  tab.array     |
0008 AREF   0007  0003          |
0009 ALOAD  0008                |
0010 SLOAD  #4    T             |
0011 ADD    0010  0009          |
0012 ADD    0003  +1            |
0013 LE     0012  0001          |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6   CRI           | 0001
0002 LE     0001 +2147483646   | 0002
0003 SLOAD  #5   CI            | 0012
0004 SLOAD  #2   T             | 0004
0005 FLOAD  0004 tab.asize     | 0005
0006 ABC    0005 0001          | 0006
0007 FLOAD  0004 tab.array     | 0007
0008 AREF   0007 0003          |
0009 ALOAD  0008               |
0010 SLOAD  #4   T             |
0011 ADD    0010 0009          |
0012 ADD    0003 +1            |
0013 LE     0012 0001          |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI              | 0001
0002 LE     0001  +2147483646      | 0002
0003 SLOAD  #5    CI               | 0012
0004 SLOAD  #2    T                | 0004
0005 FLOAD  0004  tab.asize        | 0005
0006 ABC    0005  0001             | 0006
0007 FLOAD  0004  tab.array        | 0007
0008 AREF   0007  0003             | ===> AREF  [0007][0003]
0009 ALOAD  0008                   |
0010 SLOAD  #4    T                |
0011 ADD    0010  0009             |
0012 ADD    0003  +1               |
0013 LE     0012  0001             |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI         | 0001
0002 LE     0001  +2147483646 | 0002
0003 SLOAD  #5    CI          | 0012
0004 SLOAD  #2    T           | 0004
0005 FLOAD  0004  tab.asize   | 0005
0006 ABC    0005  0001        | 0006
0007 FLOAD  0004  tab.array   | 0007
0008 AREF   0007  0003        | ===> AREF   0007  0012
0009 ALOAD  0008              |
0010 SLOAD  #4    T           |
0011 ADD    0010  0009        |
0012 ADD    0003  +1          |
0013 LE     0012  0001        |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 |
0010 SLOAD  #4    T              |
0011 ADD    0010  0009           |
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI         | 0001
0002 LE     0001  +2147483646 | 0002
0003 SLOAD  #5    CI          | 0012
0004 SLOAD  #2    T           | 0004
0005 FLOAD  0004  tab.asize   | 0005
0006 ABC    0005  0001        | 0006
0007 FLOAD  0004  tab.array   | 0007
0008 AREF   0007  0003        | 0015 AREF   0007  0012
0009 ALOAD  0008              | 0016 ALOAD  0015
0010 SLOAD  #4    T           |
0011 ADD    0010  0009        |
0012 ADD    0003  +1          |
0013 LE     0012  0001        |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI          | 0001
0002 LE     0001  +2147483646  | 0002
0003 SLOAD  #5    CI           | 0012
0004 SLOAD  #2    T            | 0004
0005 FLOAD  0004  tab.asize    | 0005
0006 ABC    0005  0001         | 0006
0007 FLOAD  0004  tab.array    | 0007
0008 AREF   0007  0003         | 0015 AREF   0007  0012
0009 ALOAD  0008               | 0016 ALOAD  0015
0010 SLOAD  #4    T            | 0011
0011 ADD    0010  0009         |
0012 ADD    0003  +1           |
0013 LE     0012  0001         |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI              | 0001
0002 LE     0001  +2147483646      | 0002
0003 SLOAD  #5    CI               | 0012
0004 SLOAD  #2    T                | 0004
0005 FLOAD  0004  tab.asize        | 0005
0006 ABC    0005  0001             | 0006
0007 FLOAD  0004  tab.array        | 0007
0008 AREF   0007  0003             | 0015 AREF   0007  0012
0009 ALOAD  0008                   | 0016 ALOAD  0015
0010 SLOAD  #4    T                | 0011
0011 ADD    0010  0009             | 0017 ADD    0011  0016
0012 ADD    0003  +1               |
0013 LE     0012  0001             |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI          | 0001
0002 LE     0001  +2147483646  | 0002
0003 SLOAD  #5    CI           | 0012
0004 SLOAD  #2    T            | 0004
0005 FLOAD  0004  tab.asize    | 0005
0006 ABC    0005  0001         | 0006
0007 FLOAD  0004  tab.array    | 0007
0008 AREF   0007  0003         | 0015 AREF   0007  0012
0009 ALOAD  0008               | 0016 ALOAD  0015
0010 SLOAD  #4    T            | 0011
0011 ADD    0010  0009         | 0017 ADD    0011  0016
0012 ADD    0003  +1           | 0018 ADD    0012  +1
0013 LE     0012  0001         |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 | 0016 ALOAD  0015
0010 SLOAD  #4    T              | 0011
0011 ADD    0010  0009           | 0017 ADD    0011  0016
0012 ADD    0003  +1             | 0018 ADD    0012  +1
0013 LE     0012  0001           | 0019 LE     0018  0001
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI         | 0001
0002 LE     0001  +2147483646 | 0002
0003 SLOAD  #5    CI          | 0012
0004 SLOAD  #2    T           | 0004
0005 FLOAD  0004  tab.asize   | 0005
0006 ABC    0005  0001        | 0006
0007 FLOAD  0004  tab.array   | 0007
0008 AREF   0007  0003        | 0015 AREF   0007  0012
0009 ALOAD  0008              | 0016 ALOAD  0015
0010 SLOAD  #4    T           | 0011
0011 ADD    0010  0009        | 0017 ADD    0011  0016
0012 ADD    0003  +1          | 0018 ADD    0012  +1
0013 LE     0012  0001        | 0019 LE     0018  0001
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI         | 0001
0002 LE     0001  +2147483646 | 0002
0003 SLOAD  #5    CI          | 0012
0004 SLOAD  #2    T           | 0004
0005 FLOAD  0004  tab.asize   | 0005
0006 ABC    0005  0001        | 0006
0007 FLOAD  0004  tab.array   | 0007
0008 AREF   0007  0003        | 0015 AREF   0007  0012
0009 ALOAD  0008              | 0016 ALOAD  0015
0010 SLOAD  #4    T           | 0011
0011 ADD    0010  0009        | 0017 ADD    0011  0016
0012 ADD    0003  +1          | 0018 ADD    0012  +1
0013 LE     0012  0001        | 0019 LE     0018  0001
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
.... SNAP   [ ---- ---- ---- ---- 0017 0018 0001 ---- 0018 ]
```

```
0001 SLOAD  #6    CRI          | 0001
0002 LE     0001  +2147483646  | 0002
0003 SLOAD  #5    CI           | 0012
0004 SLOAD  #2    T            | 0004
0005 FLOAD  0004  tab.asize    | 0005
0006 ABC    0005  0001         | 0006
0007 FLOAD  0004  tab.array    | 0007
0008 AREF   0007  0003         | 0015 AREF   0007  0012
0009 ALOAD  0008               | 0016 ALOAD  0015
0010 SLOAD  #4    T            | 0011
0011 ADD    0010  0009         | 0017 ADD    0011  0016
0012 ADD    0003  +1           | 0018 ADD    0012  +1
0013 LE     0012  0001         | 0019 LE     0018  0001
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
.... SNAP   [ ---- ---- ---- ---- 0017 0018 0001 ---- 0018 ]
```

```
0001 SLOAD  #6    CRI         | 0001
0002 LE     0001  +2147483646 | 0002
0003 SLOAD  #5    CI          | 0012
0004 SLOAD  #2    T           | 0004
0005 FLOAD  0004  tab.asize   | 0005
0006 ABC    0005  0001        | 0006
0007 FLOAD  0004  tab.array   | 0007
0008 AREF   0007  0003        | 0015 AREF   0007  0012
0009 ALOAD  0008              | 0016 ALOAD  0015
0010 SLOAD  #4    T           | 0011
0011 ADD    0010  0009        | 0017 ADD    0011  0016
0012 ADD    0003  +1          | 0018 ADD    0012  +1
0013 LE     0012  0001        | 0019 LE     0018  0001
                                0020 PHI    0012  0018
                                0021 PHI    0011  0017
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */

  return fleft->op2;
}
```

```c
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */
  /* What if fleft is not invariant? */
  return fleft->op2;
}
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */
  PHIBARRIER(fleft);
  return fleft->op2;
}
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */
  PHIBARRIER(fleft);
  return fleft->op2;
}
```

DCE
LOOP
SPLIT
SINK

assemble

```
asm_guardcc(as, CC_E);
emit_rr(as, XO_TEST, RID_RET, RID_RET);
```

```
asm_guardcc(as, CC_E);
emit_rr(as, XO_TEST, RID_RET, RID_RET);
/* looks a bit strange?  */
```

```
asm_guardcc(as, CC_E);
emit_rr(as, XO_TEST, RID_RET, RID_RET);
/* assembled backwards!  */
/* test rax, rax; je ... */
```

## linear scan

THE END

• • •

tab.fld

```
0003 int FLOAD 0002 tab.hmask
0004 int EQ    0003 XXXX
0005 p32 FLOAD 0002 tab.node
0006 p32 HREFK 0005 "fld" @YYYY
0007 num HLOAD 0006
```

```
cmp dword [rdx+0x1c], XXXX
jnz ->0
mov ecx, [rdx+0x14] ; tab.node
mov rdi, 0xfffffffb00052de0 ; "fld"
cmp rdi, [rcx+YYYY]
jnz ->0
lea eax, [rcx+0x18]
cmp dword [rax+0x4], 0xfffeffff
jnb ->0 ; is num?
```

# OOP?

```lua
local M = {}
function M:getFld()
  return self.fld
end

local s = setmetatable({fld = 1},
                        {__index = M})
local sum = 0
for i = 0, 100 do
  sum = sum + s:getFld()
end
```

```
0003     p32 HREF   0002  "getFld"
0004 >   p32 EQ     0003  [0x00042458]
0005     tab FLOAD  0002  tab.meta
0006 >   tab NE     0005  NULL
0007     int FLOAD  0005  tab.hmask
0008 >   int EQ     0007  +1
0009     p32 FLOAD  0005  tab.node
0010 >   p32 HREFK  0009  "__index" @1
0011 >   tab HLOAD  0010
0012     int FLOAD  0011  tab.hmask
0013 >   int EQ     0012  +1
0014     p32 FLOAD  0011  tab.node
0015 >   p32 HREFK  0014  "getFld" @0
0016 >   fun HLOAD  0015
0017 >   fun EQ     0016  y.lua:4
... fld load here ...
```

```
0003      p32 HREF   0002  "getFld"
0004 >    p32 EQ     0003  [0x00042458]
0005      tab FLOAD  0002  tab.meta
0006 >    tab NE     0005  NULL
0007      int FLOAD  0005  tab.hmask
0008 >    int EQ     0007  +1
0009      p32 FLOAD  0005  tab.node
0010 >    p32 HREFK  0009  "__index" @1
0011 >    tab HLOAD  0010
0012      int FLOAD  0011  tab.hmask
0013 >    int EQ     0012  +1
0014      p32 FLOAD  0011  tab.node
0015 >    p32 HREFK  0014  "getFld" @0
0016 >    fun HLOAD  0015
0017 >    fun EQ     0016  y.lua:4
... fld load here ...
```

```
0003      p32 HREF   0002  "getFld"
0004 >    p32 EQ     0003  [0x00042458]
0005      tab FLOAD  0002  tab.meta
0006 >    tab NE     0005  NULL
0007      int FLOAD  0005  tab.hmask
0008 >    int EQ     0007  +1
0009      p32 FLOAD  0005  tab.node
0010 >    p32 HREFK  0009  "__index" @1
0011 >    tab HLOAD  0010
0012      int FLOAD  0011  tab.hmask
0013 >    int EQ     0012  +1
0014      p32 FLOAD  0011  tab.node
0015 >    p32 HREFK  0014  "getFld" @0
0016 >    fun HLOAD  0015
0017 >    fun EQ     0016  y.lua:4
... fld load here ...
```

```
0003     p32 HREF   0002  "getFld"
0004 >   p32 EQ     0003  [0x00042458]
0005     tab FLOAD  0002  tab.meta
0006 >   tab NE     0005  NULL
0007     int FLOAD  0005  tab.hmask
0008 >   int EQ     0007  +1
0009     p32 FLOAD  0005  tab.node
0010 >   p32 HREFK  0009  "__index" @1
0011 >   tab HLOAD  0010
0012     int FLOAD  0011  tab.hmask
0013 >   int EQ     0012  +1
0014     p32 FLOAD  0011  tab.node
0015 >   p32 HREFK  0014  "getFld" @0
0016 >   fun HLOAD  0015
0017 >   fun EQ     0016  y.lua:4
... fld load here ...
```

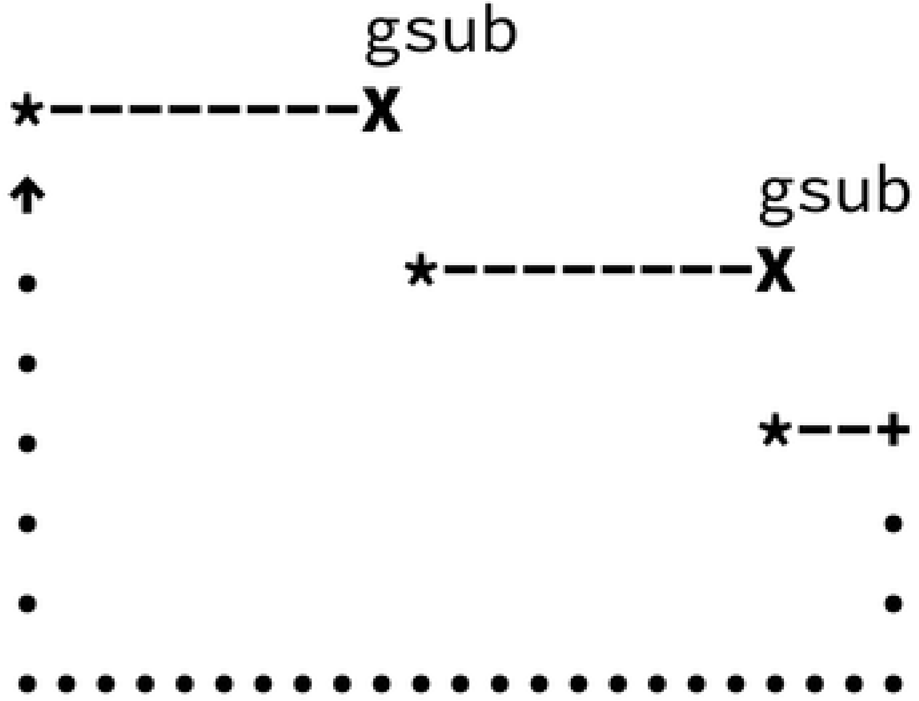
problematic if not invariant

# traces are not reentrant

[can't call lua_CFunction&stay on trace]

[though LJ2.1 has *stitching*]

```lua
local str = "abcd"
local sum = 0
for i = 0, 100 do
  str = str:gsub('a', 'z')  -- C func
           :gsub('z', 'a')  -- C func
end
```


gsub
gsub

```
         gsub
*--------X
↑                  gsub
.         *--------X
.
.                  *--+
. state transfer via  .
. interpreter state   .
.......................
```

## builtin library?

## builtin library?

[need to record manually]

[LJ2.1 has LJLIB_LUA]

```
LJLIB_LUA(table_remove) /*
  function(t, pos)
    CHECK_tab(t)
    local len = #t
    if pos == nil then
      if len ~= 0 then
        local old = t[len]
        t[len] = nil
        return old
      end
    else
      -- ...
    end
  end
*/
```

## FFI

```
ffi.cdef [[
typedef struct { int32_t x, y; } S;
double f(S* p, size_t n);
]]
local S = ffi.typeof('S')

local arr = ffi.new('S[?]', 2)
arr[0] = S(1, 2)
arr[1] = S(3, 4)
ffi.C.f(arr, 2)
```

ffi objects have frozen metatables

[see issue #41 for normal tables]

```lua
ffi.cdef[[
typedef struct { int32_t x, y; } S;
]]
local M = {}
function M:getX() return self.x end
local S = ffi.metatype('S', {__index=M})
local s = S(1,2)

local sum = 0
for i = 0, 100 do
  sum = sum + s:getX()
end
```

```
0003      u16 FLOAD  0002  cdata.ctypeid
0004 >    int EQ     0003  +XXXX
0005      p64 ADD    0002  +YYYY
0006      int XLOAD  0005
```

no table probing!

## side-traces

## side-traces

[not all values are carried inside]

[rejoins at the trace entry]

… one more thing

```lua
local function faster(arr, n)
  local sum = 0
  for i = 1, n do
    sum = sum + arr[i]
  end
  return sum
end

local function slower(arr, n)
  local sum, i = 0, 1
  while i <= n do
    sum = sum + arr[i]
    i = i + 1
  end
  return sum
end
```

## What I learned from LuaJIT

## ELEGANCE IS A DOUBLE-EDGED SWORD

## DO NOT FEAR THE PREPROCESSING

## USERS DON'T UNDERSTAND WHAT IS FAST

## PERFORMANCE IMPLICATIONS OF TRACING ARE NONTRIVIAL

## SEARCH FOR THE BALANCE

MAKE YOUR OWN RULES

## THANK YOU!